ARPA ORDER NO.: 189-1
7P10 Information Processing Techniques

R-2134-ARPA
December 1977

Framework and Functions of the "MS" Personal Message System

David H. Crocker

A Report prepared for

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under Contract No. DAHC15-73-C-0181.

Published by The Rand Corporation

ARPA ORDER NO.: 189-1
7P10 Information Processing Techniques

R-2134-ARPA
December 1977

Framework and Functions of the "MS" Personal Message System

David H. Crocker

A Report prepared for

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Rand
SANTA MONICA, CA. 90406

PREFACE

A coordinated set of programs for minicomputers in the Digital Equipment Corporation PDP-11 series is being developed by The Rand Corporation as part of its research for the Information Processing Techniques Office of the Defense Advanced Research Projects Agency (ARPA). This computer software will enable a user to perform such tasks as text manipulation, "reminder" functions, rule-directed "user agent" functions, and manipulation of electronic mail.

This report describes the design of one such program--the "MS" message system. Early electronic mail systems have existed on the larger computers. MS incorporates and expands upon many of the functions and concepts of such systems within an integrated package, using the Unix operating system, for users of PDP-11 minicomputers.

The report should be of interest to users and designers of computer-based communication network message systems. Familiarity with the Unix operating system, although not critical to an understanding of the text, would be helpful to most readers. This document is not intended to serve as a user's guide. As specific interfaces with human operators are constructed for MS, specific user's guides are being written.

SUMMARY

One of the earliest and most popular applications of the ARPANET computer communications network has been the transfer of text messages between people using different computers. This "electronic mail" capability was originally grafted onto existing informal facilities; however, they quickly proved inadequate. A large network greatly expands the base of potential communicators; when coupled with the communication convenience of a message system, there results a considerable expansion to the list of features desired by users. Systems which have responded to these increased user needs have resided on medium- and large-scaled computers.

The Unix operating system, which runs on DEC PDP-11 minicomputer hardware, has not benefited from recent advances in network mail technology. This report describes some of the issues surrounding the design of such technology and specifies a system which transfers it to Unix. In the form specified, MS is intended to be an interim facility, having maximal utility for three to five years. In addition, the system is expected to provide a base for future generations of Unix message systems.

The MS environment consists of several pieces of software to compose, transmit, receive, review, and manipulate messages. Messages reside in file "folders" and may contain any number of fields, or "components". The user can arbitrarily name, create, and modify these components. In particular, a draft message is provided for composing new mail and modification of old messages is allowed. The user is

thereby given a relatively homogeneous and unrestricted environment for manipulating mail, although facilities for data-base management (filing and cataloging) and for personal tailoring of system behavior are relatively primitive.

The specifications in this report differ from the style of most system specifications; normally, either the way the system is to appear at its interface to human users, or else the range of primitive operations and "data objects" available is defined. Although they have more of the flavor of an interface description, the specifications here do not describe the precise way in which users formulate requests. That is, the functions, to be made available to human users, are described; however, the command language for invoking those functions is not. The reason for this idiosyncratic specification style is that several very different command interfaces are being constructed, and it is hoped that specification at this level will simplify the task of implementing them.

A number of features, normally reserved for user interfaces, are provided by the basic MS system; it is intended that these features will facilitate the design of interfaces to adequately respond to psychological aspects of using interactive computer systems and, in particular, that the appearance of the system will conform to typical users' cognitive models of a message-processing environment. This report includes discussion of these issues.

ACKNOWLEDGMENTS

This system description has benefited from the support of a large number of people. Many of the ideas in this document have been freely incorporated from those instantiated in existing systems, mentioned in the Introduction, and from a continuing set of discussions about message systems, which has taken place among more than seventy researchers distributed around the country and using ARPA Network message systems. In particular, Robert Anderson, Carl Sunshine, Stockton Gaines, James Gillogly (all of Rand), Steven Zucker (formerly with Rand), Stephen Kent (a summer Consultant from MIT), David Farber (University of Delaware), John Vittal (Bolt Beranek & Newman), and Kenneth Pogran (MIT) have reviewed and enhanced the original system specifications. William Crosby is the system's primary implementor, with Steven Tepper implementing the network, address, and initial command-specification software; both have been diligent at finding inconsistencies in and omissions from the original specifications. Sally Wallace, Grace Carter, and Lynn Anderson of Rand, and Cathy Koerner, formerly with Rand, were tolerant subjects for informal experiments conducted to select function characteristics for the system.

CONTENTS

FIGURES

I. INTRODUCTION

BACKGROUND

Time-shared computers typically have a system which allows their users to pass informal messages among themselves. As long as a computer is not connected to other computers, its community of users remains relatively small and geographically local, and its message system tends to remain relatively simple and used only for terse, infrequent communications.

The advent of the ARPA computer communications network (ARPANET) (Roberts & Wessler, 1970; Crocker, Heafner, Metcalfe & Postel, 1972) has dramatically changed such usage patterns. Message systems, coupled with a large network, result in a substantial pool of potential users who can obtain rapid delivery of messages (relative to the U.S. Postal Service) and an asynchronous interaction style which allows composition, transmission, receipt, and perusal at the convenience of each participant. The telephone allows more rapid delivery of information and an interaction style which often leads to greater effective bandwidth, but it requires participants to schedule contacts. It is therefore not surprising that a computer-based message system can fill an important niche in human communication and has become extremely popular with its community of users, replacing a substantial portion of normal mail and telephone activity.

Initially, the network communication capability was simply grafted onto existing intra-machine message facilities; however, growth in use of the facilities has led to considerable expansion of the list of features desired by users (Uhlig, 1977). For an introduction to the context and economics of electronic mail, see Vezza (1975), Vezza and Broos (1976), and Panko (1976).

The first integrated ARPANET-based software to gain wide acceptance for this type of "automated office" application was the BANANARD system (Yonke, 1975) and its successor, MSG (Vittal, 1975), written at U.S.C.'s Information Sciences Institute (ISI) for the Tenex operating system (Bobrow, Burchfiel, Murphy & Tomlinson, 1971; Myer, Barnaby & Plummer, 1971) which runs on Digital Equipment Corporation (DEC) PDP-10 hardware. MSG provides basic capabilities for creating, sending, viewing, storing, answering, and forwarding messages; its database management and message-revision functions are rather primitive. See Fig. 1 for an annotated scenario of a typical session with MSG.

Over the past several years, other message-system development efforts have begun; all attempt to provide a quantum improvement to the level of capabilities offered in MSG. Among development efforts, Stanford Research Institute's Augmentation Research Center (Engelbart, 1972), ISI (Tugender & Oestreicher, 1975), Bolt Beranek and Newman (Myer & Mooers, 1976), and MIT's Dynamic Modeling Systems project (Broos, Black & Vezza, 1975) have been most noteworthy in the ARPA community.

User first types the character after "<—"; MSG prints rest of word; may be repeated for qualifiers, such as "all messages". User's text is in boldface; comments are in the right column.

@MSG

MSG -- version of 1 April 1976

<— Headers All messages *This is a menu of my messages*

1 22 OCT To: Deutsch Topic detection literature
. . . . . .

6 27 OCT Hathaway Analogies

<— Snd msg [Confirm] Yes *Create & send message*

To (? for help): **Greep @ isd** *Primary recipients,*
cc (? for help): **Dcrocker@isd** *Secondary recipients*
Subject: **Test** *Topic of message*

Message *Text of message:*
[Complete typing message body.
↑Z to finish, ↑N to abort]
This is a test message, being used to help generate a scenario of MSG use, on Tenex, Dave.
↑Z *Indicates I am finished*

Greep at ISD -- ok *The message has been sent.*
Dcrocker at ISD -- ok

<— Type 6 *Print message #6 on my terminal*

(msg. #6, 1500 chars)
Date: 27 OCT 1976 1218-PDT
To: Header-People at MIT-MC
From: Hathaway at AMES-67
Subject: Analogies

<<text of message from Hathaway>>

<— Answer message: 6 *I want to respond to message number 6*

6 27 OCT Hathaway Analogies

[Complete typing message body. *Addresses automatically filled out; I just write my response text*

↑Z to finish, ↑N to abort
<< text of my response to Hathaway>>
↑Z

Hathaway at AMES-67 -- ok
<— Delete 6 *Throw message away*

<— Headers Deleted messages

*6 27 OCT Hathaway Analogies
*9 6 NOV To: Pogran Re: stuff

<— Exit and update old file [Confirm] **Yes**
Good-bye

Fig. 1—Sample MSG session

The Unix time-sharing system (Ritchie & Thompson, 1974), which runs on DEC PDP-11 minicomputer hardware, has not benefited from these later developments and has had only an informally-developed system with capabilities at about the level of BANANARD. Rand has undertaken the design and development of a more complete and integrated message system, transferring proven message-system technology onto a minicomputer. This system, called MS (pronounced "mizz"), was to provide capabilities at least equivalent to those of the MSG system and it has been designed to evolve to the level of state-of-the-art systems. The initial version of MS was to have a projected life-span of three to five years.

MS became operational at Rand at the end of 1976 and received limited distribution to other ARPA-project Unix machines by summer 1977. The system appears to provide a better framework for growth than was expected. It has been continually modified, as experience has uncovered deficiencies in the original design; no major problems have been encountered in effecting these changes.

Due to the evolutionary nature of MS, this document cannot be a definitive specification of all of the system's features. Therefore, a portion of the text is devoted to extensive explanation of the perspective with which design decisions are being made. Some of the perspective is the result of constructing MS after the ISI, BBN, MIT and SRI systems and reflects various of their characteristics. Since a message system is a complex environment, it is not possible to list those reflections accurately or completely. Attention also has been given to the importance of psychological and environmental factors in the use of interactive computer systems. While such social issues can be characterized globally, and the resulting basic design decisions can be discussed, it is not possible to explain all ways in which MS has been affected by these considerations.

FRAMEWORK FOR USING MESSAGE SYSTEMS

As suggested by the sample MSG session in Fig. 1, messages on the ARPANET can be characterized as "memos". They are relatively structured and, since they must be represented in a single coding system (the ASCII character set), can have only one typeface, size, and color -- though it should be noted that the system or terminal used by the receiver of a message can (at least potentially) choose the face, size and color. At present, it is not possible to send drawings, facsimile, speech, or structured text. Such restrictions make ARPANET mail appropriate for most intra-organization and some inter-organization communications. The ARPANET message environment is currently biased towards use as an informal communication mechanism but is being adapted for more formal activity. In normal offices, this combination represents a substantial portion of paper-based communication and can be expected to result in a considerable amount of computer-based mail-processing. Experience with ARPANET message activity by managers bears out this expectation. Even with somewhat restricted machinery, such as terminals which print at only thirty

characters a second, it is not uncommon for a user to process twenty to fifty messages a day.

It appears that most users of computer message systems are extremely intolerant of idiosyncratic system behavior. They wish to use the system to communicate with other humans and do not want the computer--the communication medium--to intrude on that process. Curiously, this fact tends to apply even to those with a high degree of sophistication about computing.

This phenomenon also occurs with users of certain other tools, such as text editors. These systems augment rather basic human communication activities and require a kind of "intimate interaction," which can be characterized as sustained request/response sequences with most transactions involving conceptually simple actions by the computer and requiring between one-half and two seconds to complete. (Carbonnell, Elkind & Nickerson, 1968). Much of this activity is characterized as requiring "routine cognitive skill" (Card, Moran & Newell, 1976).

Since the system is to be used for communication which is exemplified in older and heavily-exercised technology, it is assumed that users have an extensive conceptual model of the communication domain. It is further assumed that a system which performs in ways which deviate from that model will be viewed as "idiosyncratic" and impeding the efforts of the user. Problems occurring during this sort of interaction can be expected to be as irritating as having a pen which leaks or a typewriter with keys that jam. Therefore, a major design goal for MS is to provide an integrated set of necessary and sufficient functions which conform to the target user's cognitive model of a regular office-memo system. At this stage, no attempt is being made to emulate a full-scale inter-organization mail system.

AN OPERATIONAL MODEL

The scope of the MS project has not permitted empirical verification of the majority of its assumptions about the presence and characteristics of users' conceptual models for message activity. The project has had to rely upon the intuitive appeal of its assumptions and the degree to which other systems seem to succeed or fail in terms of their conformance and deviation from that model. Work by Heafner (1976), Heafner and Miller (1976), and others suggests that the model does exist and can be characterized. Work by Brown and Klerer (1975), Kennedy (1975), Walther, (1973) and Carlisle (1974) suggests that the degree to which a system conforms to users' expectations and abilities will have a significant effect upon their use of that system.

Because the system processes structured "memos", the basic unit of manipulation is taken to be the "component". A hierarchy is formed by having memos (or "messages") consist of collections of particular components, and "folders" as collections of particular messages. Messages have some common components, such as "To", "From", and "cc", but

individual messages may have additional components with unique names. In addition, common names vary between contexts, such as the difference between business and military terminology. MS attempts to give users complete control over the naming and accessing of components.

A message assumes an identity as soon as any of its text is created. Over the life of a message, various actions may be performed on it. Some of these actions occur more commonly at certain phases than at others; however, this generally does not mean that these actions are prohibited during other phases. For example, a message is often revised before it is sent and rarely revised afterwards; but some revisions may occur, as when recipients make notations in its margins or when one recipient is part of a message "coordination" process and charged with passing a revised version of the message onto others.

Within an office environment, messages typically arrive at a person's "inbox", are viewed and perhaps acted upon, and are then filed into an appropriate folder which contains related messages. Later, the person may wish to take other actions relating to the material in the folder. All of this activity occurs on the person's desk. Several folders may be open at one time.

Two of the more common actions people take are responding to a message and forwarding a copy of it to others. In both cases, material in the original message determines portions of the new message. For responses, the title ("Subject") and the names of the originator and recipients are used; and for forwarded messages, only the name(s) of new recipient(s) must be added. Another common action is the creation of a new message for a third party.

When a comparison is made between the way these actions are normally performed in an office and the way they are performed using some existing computer-based message systems, several issues of operational styles surface:

1. Messages which are being created ("draft" messages) must be treated in fundamentally the same way as messages which have already been sent (and received);

2. A message may have "draft" status for an extended period of time, rather than being sent immediately after creation; and

3. Several draft messages may exist at one time.

The first point implies a more general issue: humans often do not make distinctions in the same ways as computers. For efficiency, a computer might handle a draft message differently than it handles "old" messages or that it might copy some kinds of text differently than other kinds. Humans, however, are generally not conscious of the conceptual distinctions which lead to these differences in handling.

Imposing such distinctions upon users is another case in which the system will probably be classed as idiosyncratic and counterproductive.

The final way in which MS attempts to conform to users' expectations is in the vocabulary used to describe and invoke its processing. Concern for this level of detail has been questioned, on the theory that humans are quite good at learning new terms and, in fact, they are not consistent in their own use of vocabulary. That is, there probably does not exist a set of terms which is consistent among users and, even if there is, using that set rather than another will probably not greatly affect a user's performance with, or attitude towards, a message system.

In the belief that computer-oriented users and designers cannot be used as references for testing the presence and nature of such vocabulary in the potential user population, several informal experiments were conducted. Subjects were secretaries who had little or no experience using computers. In each case, relatively neutral language was used to explain a typical office situation which required use of a single word for referencing a particular object or action. The subject was then asked what word or symbol was most appropriate in that situation. In most cases, subjects immediately had a term they thought best and the terms were relatively consistent among subjects.

For example, a message being created is called a "draft"; the structured part of a memo is called the "headers"; and placing a message into a folder is called "filing". While such terms may seem trivially obvious, many message systems use terms which do not even approximate those offered by subjects. In fact, some systems use terms which have significantly different implications. For example, to "put" a message somewhere means that the original message changes location; however in some systems, the word is often used to cause an action which only places a copy of the message somewhere. It should be noted that, as Heafner (1976) has demonstrated, acquiring this sort of empirical data, in a methodologically valid manner, is relatively easy and inexpensive.

It is difficult to substantiate the claim that use of the most predictable vocabulary actually affects users' performance and attitudes. Except for that cited earlier, little research has been done to test the idea. It is noteworthy, however, that subjects in the informal experiment often reacted quite strongly when queried for certain vocabulary; their choices were so well-ingrained that they could not believe there was any question about their selection. Telling them of the terms used by some computer systems often evoked laughter. It seems to the author that such a reaction establishes a mental set which is quite likely to deter users from a system and cause them confusion when dealing with it. This is particularly critical during their initial use, since they will often already have enough difficulty becoming familiar with computer-related conventions and concepts that cannot be avoided.

SCOPE OF SPECIFICATION AND IMPLEMENTATION

This document, and the style of the resulting system implementation it specifies, is a bit unusual and deserves some explanation. Most system specifications address either the human interface or the internal design -- how the system appears to human users or what data structures and function primitives are to exist. The specification for MS is at neither level, although it has more of the flavor of an interface description. In particular, the document may be viewed as specifying the human interface, minus the command language. That is, the functions, to be made available to human users, are described; but the precise way in which users formulate requests to MS is not.

The reason for this idiosyncratic specification style is that several very different command interfaces are being constructed and it is hoped that specifying the system at this level will assist interface builders in realizing and accommodating some of the user issues described above. (The concern for proper vocabulary, therefore, is more representative of a lobbying effort than of a guarantee for what is to be provided in the command interfaces.) Experience to date suggests that the construction of interfaces is, in fact, simplified.

Three general-purpose interfaces have already been constructed. The first, described in Appendix B, is intended to be similar to the basic syntax of the Unix Shell (Thompson & Ritchie, 1975), which is the program that users employ to gain access to most of Unix's capabilities. (See Fig. 2 for a sample Shell session.) This choice was made because MS is intended for use on other Unices in other environments, and having a familiar command specification style was deemed more important than providing an especially "friendly" interface. Fig. 3 shows a sample session, using the Shell-syntax interface; and Appendix D contains an extended example of using this interface. The second interface constructed emulates the Unix "mail" command and is primarily intended for use by programs to send mail to users. The third interface emulates MSG, since MSG is a de facto standard on the ARPANET, with behaviors that are already familiar to many people.

In general, it is expected that users will be provided with a single command interface to the full message system, rather than be forced to deal with two or more different systems--for example, one program for creating and sending messages and another for reading and filing them. This should not preclude additional interfaces to subsets of the system, as would be appropriate if the user only wanted to send a message quickly. However, such programs should be strict subsets of the full system.

The level of the MS project effort has also had a major effect upon the system's design. To construct a fully-detailed and monolithic message processing environment requires a much larger effort than has been possible with MS. In addition, the fact that the system is intended for use in various organizational contexts and by users of differing expertise makes it almost impossible to build a system which responds to all users' needs. Consequently, important segments of a

The Shell types a "%", to indicate that it is ready for the user to enter a command. Commands are specified entirely before the system attempts to perform them; boldface text is typed by the user and comments are in right column:

```
% dir                                                    What files have I when
                                                         created; how large; etc.?
total 224
-rw-r--r--   1 dcrock   10B  Nov 5   14:15   Nov 5   14:10 gnome.proto
-rw-r-r--    1 dcrock   10B  Nov 5   13:49   Nov 5   13:49 gnome.proto.bak
-rw-r-r--    1 dcrock    3B  Nov 5   13:12   Nov 5   13:12 msg.proto
-rw-r-r--    1 dcrock    5B  Nov 5   13:12   Nov 5   13:05 msg.proto.bak

% rm *.bak                                              Remove files with names
                                                         ending with ".bak".

% cp gnome.proto > gnome.pr                             Make a copy of a file,
                                                         using a shorter name.

% who                                                    Who is using the system now?

bjg        tty0  Nov  5     13:33
mother     tty1  Nov  5     12:55
herb       ttyh  Nov  5     12:20
dcrockerttys  Nov  5     13:25
andersonttyw  Nov  5     14:10

% ms                                                    Start the message system

MS:  1-NOV-76
   <<   an ms session; then . . .>>
> quit                                                  Note the ">" instead of
                                                         "%", indicating use of a
                                                         different program.
%                                                       We are back to the Shell
```

Fig. 2—Sample Shell Session

Shows facilities similar to MSG, except for Draft message, which MSG does not allow user to revise conveniently; boldface text is typed by the user:

```
% ms                                          User commands follow each
                                              ">" at beginning of lines
MS:  1-NOV-76

> scan all

     1 <= (321)    25 Oct 76  farber      Name change for system
.....
—    17    (209)    5 Nov 76  DCROCKER Test
                                              The "—" means that I have not
> show 17                                     yet seen this message

(Message 17, 209 bytes—
Date:  5 Nov 1976 1249-PST
From: DCROCKER at USC-ISI
Subject:  Test
To:  Greep at ISD
cc:  DCrocker at ISD

This is a test message, being used to help generate a scenario of MSG use, on Tenex.  Dave.

> reply 17

To:  Greep at ISD                             The system automatically
cc:  Dcrocker at ISD                          creates To, cc, & Subject.
Subject:  Re: [Test]                          Then it prompts the user
Additional cc:                                for additional copies and
                                              for text of response.

Input body.  End with <return> <control-D>.
It's very strange replying to a test message.  Dave
                                              Done entering message
Do you want to send the message now?  No

> show draft                                  Review contents of the
                                              draft message
To:  Greep at ISD
cc:  Dcrocker at ISD
Subject:  Re: [Test]
Additional cc:

It's very strange replying to a test message.  Dave

> send                                        I'm satisfied it's ready to be sent

Message is being processed
Processing completed and draft discarded

> quit                                        Leave MS; return to Shell
%
```

Fig. 3—Sample MS session

full message environment have received little or no attention and decisions have been made with the expectation that other Unix capabilities will be used to augment MS. For example, MS has fairly primitive data-base management (i.e., filing and cataloging) facilities and message folders have been implemented in a way which allows them to be modified by programs, such as text editors, which access them directly, rather than through the message system.

OVERVIEW OF MS DESIGN

The original mail system on Unix was judged sufficiently primitive that compatibility with it was not attempted. For example, the structuring of folders that contain messages differs. Current Unix software which utilizes parts of the Unix mail facility therefore needs to be modified to use the new and improved product. Systems which merely create and then send messages need not be modified, since the mail-command emulator interface allows creation of mail in exactly the same way as is done by the old Unix mail system.

The MS message environment consists of several pieces of software to compose, transmit, receive, review, and manipulate messages and to tailor the message environment. In addition, there are file folders* which contain messages. Messages, in turn, contain any number of components. In accordance with the user issues discussed earlier, an effort has been made to make the system as homogeneous as possible. For example, messages which are being created by the user and messages which have been received are equally accessible. Most system functions have a number of options available. To allow users to indicate which option settings they typically wish to employ, a profile is planned for each user.

Users will normally deal directly only with the Shell-invocable software (see Fig. 2) and with the folders which contain messages. The process of Composing messages entails placement of text into the various components of a draft message. For example, names and addresses go into the "To" and "cc" components and the text of the message goes into the body. This may be done repeatedly, allowing the user to employ a text editor to modify individual components. Transmission is an automatic process which packages the draft message to conform with ARPANET mail format standards (Pogran, Vittal, Crocker & Henderson, 1977) and delivers it into the mailboxes of all the indicated addressees. When receiving messages, the user may selectively Show them at the computer terminal. Further manipulation of mail can involve Forwarding copies to additional recipients, Replying to its authors, Filing for later reference, Listing copies on a printer

---

*Official names of MS functions begin with a captal letter and are underscored whenever used in this document; other official terminology is underscored when introduced. Names of message components are in quotation marks.

and/or Discarding (into the system's "wastebasket").

The messages in a folder are like a stack of messages in a normal office file folder; they are ordered and may be referenced by their index number (i.e., position in the folder). Any number of messages may be in a folder. They contain some number of components, most of which may consist of arbitrary strings of text. In some situations, batches of messages may be referenced; special labels are allowed for specifying them. At any given moment, the system has a current folder and a current message which are under scrutiny. (The standard folder is the inbox). Having them keeps the user from being forced to specify a folder and message index for every function. Contrary to most other message systems, most MS functions can operate on any component of any message in any folder, without requiring the user to respecify the current message or folder. Some functions cause the index of the current message to be changed; these are indicated in the functions descriptions, in Section III. When the user invokes MS, the current message is set to be just before the first recent message, so that the user may conveniently sequence through recently-arrived mail, or to the first message in the folder if there is no new mail.

When a user issues a Shell command to start the message system, the standard action will typically be to Open a folder, where this folder will usually be the user's primary folder, the inbox. (See Fig. 3.) This folder is structured like any other file which contains mail, except that new mail is placed there by the Unix mail delivery system. Current specifications call for mail to be delivered only to this folder; however, a later version may allow incoming mail to be diverted automatically to other folders, as might be appropriate to activities such as teleconferencing.

Modifications to processed messages (i.e., mail which has already been sent or received by the user) are allowed; however, in some cases, such modifications may cause the system to take note. Such exception-taking is intended only as a safety feature, as described below.

The system maintains a draft message, in its own folder. A message with "draft" status has not yet been Sent and is subject to more modification than other messages and therefore is not subject to normal access checking by functions. Current specifications allow only one draft message at a time; however there appear to be no problems in eventually permitting an arbitrary number of them. When a message is sent, a unique message-id, a timestamp, and the name of the sender are affixed if necessary.

The user can arbitrarily name, create and modify components. "To", "cc", and "Subject" are common components, but others are possible. For example, MS has a simple reporting mechanism, which allows users to send comments and complaints to the MS support staff. It automatically fills out the destination addresses and then prompts the user for the report. It also creates a component called "MS-Version" which allows the support staff to know what version of the system the

user had. Such a component will not occur elsewhere, and users are given equally unlimited creative license to formulate their own component names.

Note that no program need know the names of all possible components. To facilitate user specification and manipulation, command interfaces typically maintain a list of the common component names, and the basic system is familiar with the required "Sender", "Message-Id", "Timestamp", and "To" components, as well as "cc", "fcc" (file carbon copy), and "Subject", for the draft. Contents of these are verified or created by the system. With the exception of the first three of these components, all components of all messages may be modified, as described above.

Finally, any component may be passed to a program for manipulation. Formatting and typographical-error detection are two system-known programs. Others may be added, such as comparison of two versions of a message.

II. SYSTEM FRAMEWORK

MESSAGE FOLDERS

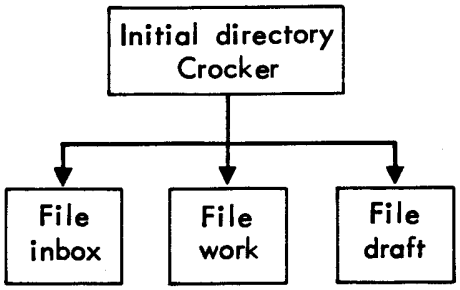
Unix organizes stored data files in a hierarchical fashion. Indexes to files and other subordinate indexes are called directories. The primary directory usually may be thought of as a filing cabinet. In a typical case, the secondary directories may be thought of as the drawers in the cabinet, and they may contain data files. Other organizational styles are possible and may become quite complex, as demonstrated by example C in Fig. 4 The simplest organization is to have only one directory and keep all files in it. Whatever the case, the user begins each session with Unix "looking at" an initial directory. If this directory contains another directory, called mail, then various standard MS files or folders are placed there. Otherwise, these folders are placed directly into the initial directory. Currently, standard folders are inbox, draft, and msreport and backup folders for draft and msreport; msreport is used by the Report function and will not be necessary when MS allows multiple draft messages.

MS folders actually consist of two Unix files. One is a clear, readable text file, organized in a fashion conforming to the ARPANET standard syntax (Pogran, et al. 1977). The second is a parallel file containing structure, status, and history information. Simple strings of special characters are used to separate messages in the "clear-text" file. If a message's format is violated, recovery is then quite simple, and the structural information in the parallel file is generally redundant and easily reconstructed. The structural information allows the system to manipulate messages and components efficiently.

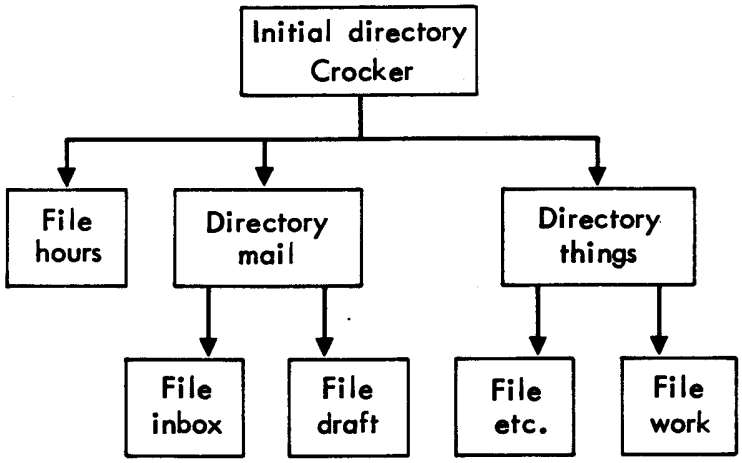
This approach is in accordance with the concern for integrating MS into the general Unix environment and for allowing the user to have unrestricted access to messages, through other Unix word-processing tools which are already familiar, such as text editors and Shell commands. Separating structural information from the data also makes it convenient to have multiple "perspectives" (or indexes) to the data. The parallel file is normally hidden from the user so that he must only deal consciously with "real" message files.

Most text-transferring functions preserve the text in its structured, processable form. The List, Scan, and Show functions are notable exceptions and do not move the information in a form compatible with further processing as a message, since they completely reorganize the text into a single string. The Copy and Map functions can also perform this alteration, under certain circumstances.

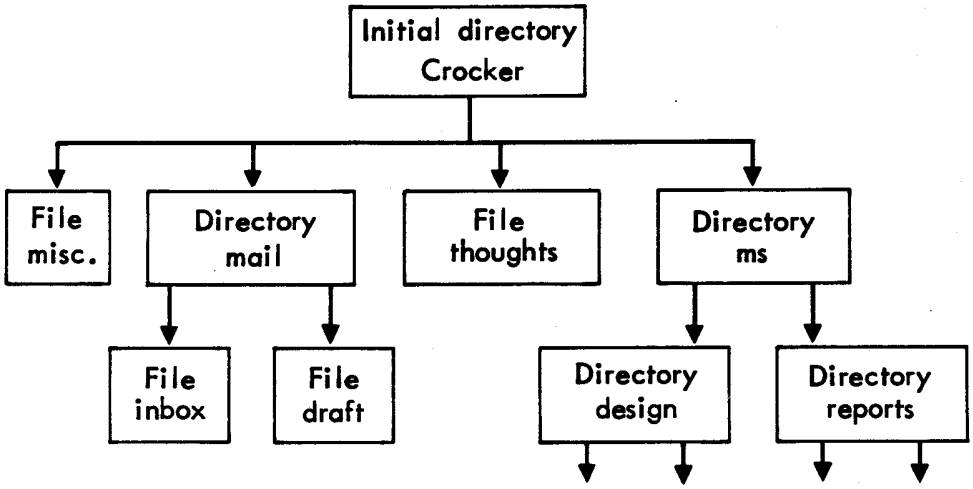
As more systems come to manipulate message files automatically, the environment will have to distinguish between activity by humans and activity by their software agents. An example of this problem occurs when another computer program checks the inbox for certain types of mail, but the human still wants to be notified of new mail. Simply checking the length of the inbox file, or when it was last



A. Simple directory, with no sub-directories



B. Directory with 1 level of sub-directories
(like cabinet with drawers)



C. Complex directory structure, with several levels

Fig. 4—Examples of file and directory

read, will therefore not provide an accurate indication of when the human last looked at the file. MS provides a solution to part of the problem: a folder may be opened with a passive status, so that no permanent actions can be performed on the folder. This capability allows automata to peruse and copy the contents of a folder, without leaving a trace of their activity in it.

MESSAGE COMPONENTS

As mentioned in the Introduction, a list of common component names is generally maintained and is used for defaults with certain functions; but such defaulting is only for convenience. At all times, MS allows modification to this list by the user, either through the Profile or at the time of creating specific components.

One of the pieces of information the system keeps about known component names is whether they are used to specify addresses. Including such a component in a message causes the message to be sent to those listed in the component. The user is able to control whether the contents of that component are included in the copy of the message sent to:

1. All recipients of the message; or
2. Other recipients named in that list; or
3. Only the author(s) copy.

This curious feature is derived from the concept of the blind carbon copy; the decision to provide so general a facility is due to a discussion with Stephen Crocker of ISI, during which the variety of distribution conventions followed by different organizations became evident. Rather than impose a single style of distributing information about who receives a message, MS lets individual users decide. The Profile allows users to alter which components are candidates for containing addresses (to be interpreted by the mail-sending process) and to alter the inclusion settings described above. In the case of the third option, a recipient's copy will show only his/her name in the component.

A more general facility would consider components to have a "data-type", with various attributes. For example, the above case would be of data-type "address" with a "distribution" attribute.

The system also allows specification of component equivalences. That is, a component name may be equivalent to some "generic" name, as in the case of "Action-to" being equivalent to "To". This facility is necessary due to the amount of variety found on the ARPANET, in the (justifiable) absence of complete naming standards. The author favors this variety, since it is the only significant control the sender can have on message appearance and the labels often have differential import, as with military versus business memo terminology.

MESSAGE CREATION

Normally, the draft message always exists, and is in a standard message folder so that creating new message text, modifying it, and then sending it when ready can be done in a fairly natural and user-controllable manner. The Compose, Ned, Ed, Correct, and Format functions, in particular, are provided to facilitate the process, but the user may easily follow different creation paths with other tools.

As described earlier, three pieces of information (and possibly more, later) are not completely controllable by the user: message creator name, message transmission date, and message identification tag. The default is for the system to place the first two pieces of information into the From and Date components, respectively. If the user explicitly manipulates one, then its backup component (Sender or Timestamp) is created. Neither the backup nor message-tag components may be modified by the user.

While typing text into a component, users often need to be able to indicate places for other text to be inserted from files such as those containing documents. Although such an action is not handled by the basic system, it should be a feature in most interfaces. A more general capability would allow the user also to include text from other components and from the output of programs.

TEXT TRANSFER AND STRUCTURED TEXT

By definition, the core of a mail system is its ability to transfer text. When done between people or systems, this is message transmission. Individuals spend most of their message processing time transferring text within their own environment ("office" or "desk"). It is important, therefore, that this type of "local" text transferring be easy to perform. MS attempts to provide reasonable access to the functions that are most frequently useful for transferring text in messages which are on a "desk". There is little experience with unusual text transferring capabilities, such as "cut and paste" editing, which might be desired by users of a computer-based mail system; however, discussions and experience on the ARPANET have led to the conclusion that the range of desired functions is large and as soon as users can conceptualize a function, they want it very much.

A computer-based message system, like MS, must be able to transfer fundamentally different types of text "objects", such as components, document files, and user input. This makes it very difficult to characterize a conceptual space for a single, "generic" transfer function; however MS attempts the characterization with its Map function. The function represents another attempt to direct interface builders, so that appropriate consideration will be given to the psychological aspects of system behavior. This section analyzes the transfer domain and describes its parameters, as used by Map. Actual behaviors are described in Section III, "Function Definitions."

The Map function represents an extreme attempt to provide the user with as integrated an environment as possible. It assumes that humans, in fact, are not aware of a distinction between transfers and therefore do not want to be forced to make one. Experience with early versions of MS suggests that the Map function may be overly ambitious. The Add and Copy functions are provided to facilitate a subset of the transfer functions which are performed frequently. The File function also performs a transfer function; additionally it Discards the source version.

A distinction must be made between human behavior which uses the generic transfer function and the analysis which attempts to understand it; this is similar to the distinction between "performance" and "competence" which linguists make. The former appears to be common enough, normally, to be performed subconsciously, as indicated by the lack of "awareness" cited above; however, gaining an intellectual understanding of the process appears to be quite difficult.

Due to the difficulties in understanding the generic function, it may be useful to review the domain of activity. The MS message system interacts with a number of related entities. The basic computer entity, which can be manipulated, is a string of text, usually acquired from the user's terminal or from a data file which is not part of the message system. Within the message system, these strings of text are placed into various components of messages. A collection of these components may constitute a message and a collection of messages may constitute a folder. Messages and folders are said to be "structured" because they are made of discrete sub-units. Fig. 5 shows the relationship between these entities.

Map is able to embody the several types of copying by using information about the source and destination to decide what kind of transfer to make. Because of the structural relationship between text entities, all text transferring may be viewed relative to components.

Four parameters of transferring are discernible:

1. Merging to a string: if more than one component provides source text, then whether to preserve their structural integrity, versus merging their contents into a single sequential string of text, e.g., a single component;

2. Merging to a message: if more than one message provides source text, then whether to preserve the exact structural relationships between the messages, versus mapping them into a single structure;

3. Naming: if the source is a component, then whether to preface the component's text with the component's name; and

4. Addition/Creation: whether the source is to be added to an

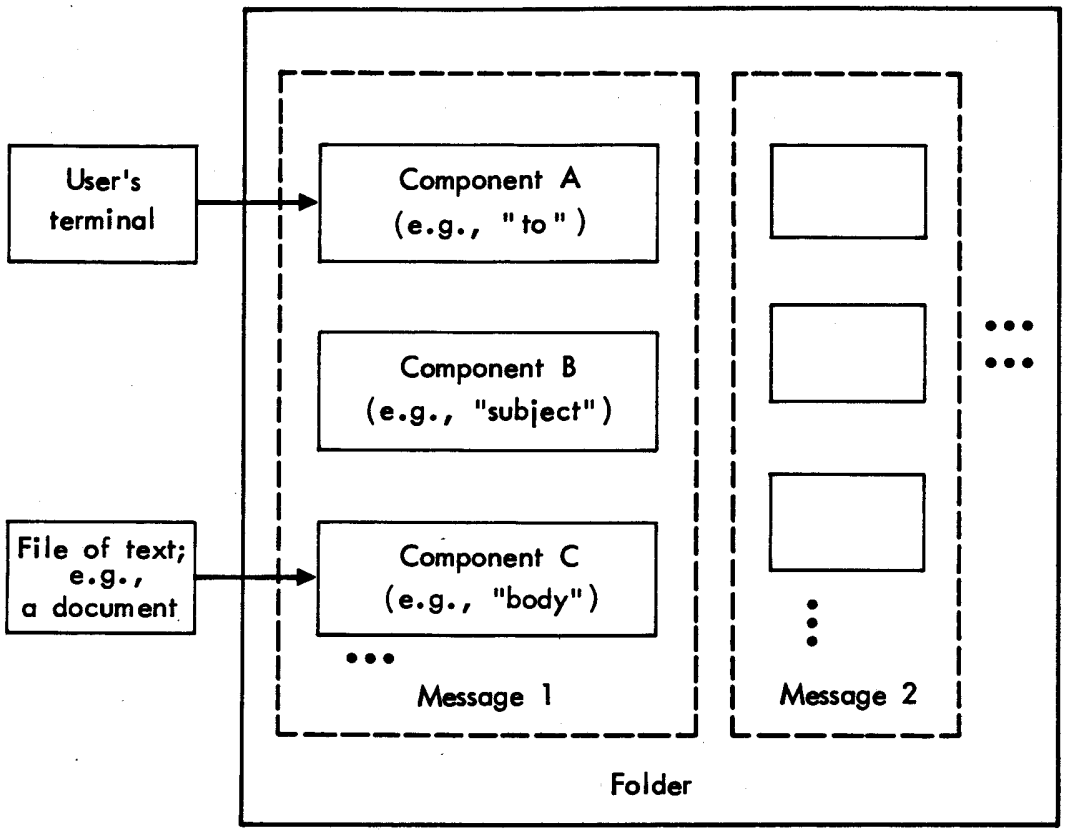


Fig. 5—Relationship between data in MS

existing structure, versus having it added to a new one.

The second alternative of the first parameter will cause transformation from internal message-system structure into clear text. The second alternative of the second parameter causes several messages to be merged into one. When the destination is merely sequential (clear) text, the third parameter determines whether the text will be "labelled" with the name of its originating component (e.g., "From", "Subject" or "To"). The fourth parameter primarily distinguishes between adding messages to a file and adding components to an existing message. Having text "added to an existing structure" can involve adding a message to an existing folder, adding components to an existing message, or adding text to the end of an existing component. In the first two cases, some new structure is also created, of course, but the focus is upon the act of adding.

If, at this point, it seems questionable that this degree of attention to such complexities is really necessary, it is worth remembering that if a person wishing to use one of these permutations does not find it available, s/he will curse the system designer for lack of foresight.

Some examples of the transfers which users are likely to want to perform, may help clarify the situation. Note that all text is transferred from a source and is appended to the end of destination(s) (components or folders), if they already exist:

1. The typical action of adding text, from a file or the terminal, to one or more components;

2. Merging the contents of existing components into a sequential string and then copying it into one or more components, as would be done when forwarding a message, by copying it into the body of a new message, or printing a message on a line-printer;

3. Copying the contents of existing components into components of the same name, in another message of the current folder; this is a kind of "forms" processing;

4. Copying one or more messages to the end of a folder, that is, filing mail for future reference.

5. Copying one or more messages to the end of a sequential string (either a component or a document file), the logical next step, after performing step 2, above;

6. Merging components of several messages into a single message and then converting to a sequential format, as a formalized combination of steps 2 and 5.

SPECIFICATION OF ADDRESSES

Experience with processing mail on the ARPANET has pointed up a number of issues pertaining to the specification of addresses. The network standard (Pogran et al., 1977) attempts to provide an adequate base for responding to the most noticeable of these. While some of them may seem trivial, they involve features and behaviors which commonly are not present in ARPANET message systems. In particular, it has been noticed that:

1. People's names are not the same as their addresses; several people may share the same inbox (address); one person may have several inboxes; and programs may wish to display a name without its associated address;

2. Mailing lists can get quite long and there needs to be a mechanism for using "named" lists;

3. To allow recipients to respond, a message often needs to carry all of its mailing list with it;

4. It is often useful to put standard lists into online files, rather than repeatedly to include their contents in messages;

5. It would be very useful to be able to send mail to folders other than a person's inbox, such as in the case of teleconferencing in which messages could automatically be grouped together, allowing the persons to peruse only conference messages.

In MS, address lists can contain the following kinds of information:

List: Mailbox at Host, Person [Mailbox at Host],
at Host, Mailbox, > filename, < filename...

Where:

List	is the optional name of the mailing list;
Mailbox	is an online reference name (usually the name of the recipient's signon directory;
at Host	gives the name of the host computer on the ARPANET containing the Mailbox;
Person	is the person's name;
filename	is the name of a file, within the user's access space;
>	indicates that a copy of the message is to be placed in the named file; and
<	indicates that the contents of the named file are to be used as an address list.

Naming the list allows a program to show only the name and not burden the user with seeing all of the names on the list. "Mailbox at Host" is the standard form of an address, and the recipient's name may be added as indicated. Referencing a computer, without a mailbox, indicates that following Mailbox references are on that computer, unless otherwise indicated.

To the extent possible during specification, addresses are checked for correctness, as soon as they are specified. For local mail, the verification is complete; however for network mail, only the name of the destination host computer can be checked. Mail which is local to the user's computer is sent through the local transmission mechanism, to avoid network transmission overhead.

It is often convenient to have a pseudonym for a person or group of people. For example, "rha" is easier to type than "anderson" and "ms-users" is easier to remember than is a list of twenty (or fifty) different people. MS provides a mechanism for using these aliases.

In MS, such aliases may be included in incoming and outgoing mail as if they were local names. When the system needs to use an address, an alias is simply replaced by a string of text and the resulting specification is treated exactly as if it was the original text. To utilize aliases on outgoing mail, MS first checks the aliases defined by users, in their Profile. If the alias is not there, MS then checks the Unix-wide alias files. If necessary, the list of known local users is then checked. This scheme allows maximal power for user-tailoring of names. For incoming mail, the search of the personal alias information is omitted. Also in outgoing mail having an alias from a personal list, the text that is shown in the message is of the text which replaces the alias. This is done so that, on other systems, legal addresses can be formed by programs that automatically generate addresses, such as is described for the Reply function in MS.

A message may contain several address lists, which can be viewed as defining different "communities". A person may be a member of more than one of these communities and may therefore appear on more than one mailing list. In order to allow independent manipulation of these lists, the person's name must be retained on each of them; however, MS will only deliver one copy of a particular message to the person.*

TRANSMISSION AND RECEIPT OF MESSAGES

All mail--both local and network--is sent and received through a special "post office" (a program in the sender's and receiver's host computer) which delivers mail to the user's primary mailbox (inbox)

---

*Such per-component manipulations appear to involve issues which are also relevant to providing multi-level security in a message system.

and may update any associated file structure information.

A feature is provided which periodically checks for recently-arrived mail in the user's inbox. Recent mail is defined as not having been accessed by the human (because they have not yet invoked or utilized the message system, since the mail arrived). The Shell will automatically check for new mail when the user returns to command level (i.e., where the "%" is typed in Fig. 2), after a fixed interval since the last check. Contrary to some current implementations elsewhere on the ARPANET, this notification does not blindly recur until the user accesses the messages; one or two notices is enough. If the user's Profile allows, the notification also includes a Scan listing (see "Function Descriptions") of the new message(s).

SEQUENCE SPECIFICATION

Within a folder, messages can be referenced by their index number (which indicates their position in the file) and a collection of messages can be referenced at one time, by using commas and dashes as connectors. They have the obvious meaning, so that the specification "1,7-9,21-2,100>99" refers to messages one, seven, eight, nine, twenty-one, twenty-two, one hundred, and ninety-nine. The angle-bracket is like dash, except that it indicates that the sub-list is in descending order.

Also, name combinations can be used to reference a particular group of components and/or one or a batch of messages. The system will recognize a number of keywords as pre-defined sequences. Fig. 6 indicates the terms that are currently available. Current specifications allow additive combinations, so that "recent, 10-15, last" will include all recent messages, the tenth through fifteenth messages in the folder, as well as the last message in it. Redundant references are not removed, so that if message fifteen is also the last message, it will occur twice. Full Boolean specification capabilities are not provided but are intended for a future version of the system.

PROFILE AND MORE STRUCTURED TEXT

This specification provides only minimal capabilities for the tailoring of MS' performance. In general, the user is able to override (Profile-set) defaults for individual executions of functions. The entire topic of individually tailorable settings is an open research question, so no attempt has been made to define an overly-sophisticated facility.

At the time of this writing, no portion of the Profile has yet been implemented. As experience developing the existing system has emphatically shown, it is highly likely that the actual form of the Profile will differ, in significant ways, from the specification in this document. In addition, discussions are underway about general "user model" features to be employed by the variety of personal-

Symbol	Type	References
(name)	(M)	Parenthesized name of a component references all messages which have a component with indicated name and contain the string of text indicated by the following argument ignoring any distinction between upper and lower case; e.g., "(from) dcrocker";
all	(MC)	(Or the character "*") All messages in the folder; or all components which are either part of the system's list or mentioned explicitly (for example, the actual component names of an existing message);
current	(M)	(Or the character ".") The message currently under scrutiny;
discarded	(MC)	Mail which has been discarded but which the janitor has not yet taken away;
draft	(M)	The single message which is being prepared.
filed	(MC)	Messages which have been copied (or filed) into another folder;
flagged	(MC)	User-specified flag for trivial subgrouping of messages;
last	(C)	The last message in the folder;
msreport	(M)	The single message which is created by the Report function.
new	(M)	Mail which has arrived since the start of the current session;
next	(M)	The first message after the "current" one; may be qualified by another keyword;
not		The complement of the next keyword;
old	(M)	Short for "not recent";
other	(C)	Components not explicitly mentioned by user or included in the system list;
previous	(C)	The first message preceding "current" one; may be qualified by another keyword;
recent	(M)	Received since user last used system;
replied	(M)	Has been marked as replied to; (also may be referenced as "answered");
seen	(MC)	Has been completely processed by a Show or List;
standard	(C)	Component names which are part of MS's list.

Fig. 6: Groups of messages (M) and components (C)

computing software being developed at Rand. It is intended that the MS Profile facility will be fully integrated into this more general user-tailoring system.

For the most part, the MS Profile facility uses the approach taken by the Hermes system, developed at BBN (Myer & Mooers, 1976), which has a relatively unorganized and large set of "switches" which can assume particular settings. A major difference is that the interface to the Profile is, itself, a series of messages, maintained in a separate folder. That is, the user alters Profile settings in exactly the same way as components of messages are altered. For the Profile "messages", a component name indicates the name of a Profile switch, and the contents of that component contain its setting. The user therefore does not need to learn any new concepts or interaction styles to be able to manipulate the Profile; and as a side benefit, the Profile settings can be shared with other users and other machines by the normal process of sending messages via the MS system. It also appears that a "message" may provide an excellent conceptual framework for coding structured information, when dealing with typical users.

Evidence from some research on memory behavior suggests that humans have short-term memory difficulty in processing "structured" information with which they are unfamiliar (e.g., Yntema & Meuser, 1960; 1962; Yntema, 1963; Yntema & Schulman, 1967). This type of information is organized into a hierarchy, or "outline" form. For example, a meal consists of several dishes. The category of dish (e.g., vegetable or entree) is one "level" in the structure while the actual dish for a particular meal (e.g., spinach or chicken) represents the "value" for that category. This defines a two-level structure, which can be extended to three levels if different meals are distinguished (i.e., meal, dish category, actual dish).

At issue is not the general ability of a person to deal with structured information, which is well documented, but rather to correctly and facilely process such information in real-time, when that information is unfamiliar to the person. The task seems to require rapid and conscious manipulation of the full information structure. Such performance requirements are generally understood to involve a mechanism known as "short-term memory," which is usually unable to hold more than approximately seven items of information at any one time (Miller, 1956). To circumvent this limit, people "chunk" information into sub-units, thereby defining the type of "outline" form described above. When the information is familiar to a person, knowledge about its structure is already stored into the infinite-capacity "long-term memory," so that s/he tends to have little difficulty in accessing arbitrary information in the structure. However with unfamiliar information, excessive hierarchization appears to overload humans with the details of the structure itself. An example of the difficulty is the number of preceding conversational contexts people can easily remember when they are repeatedly interrupted. People often are unable to remember what was being discussed only one context before the current one.

For the purposes of defining Profile switches, a three-level structure, often described as consisting of objects, attributes, and their values--embodied in MS as messages, components, and their contents--provides a reasonable compromise between the competing constraints. In addition, the concept of a message, with components, is already familiar to people and will become more familiar as they use message systems; so they should not need to learn any new concepts to manipulate a Profile which is organized as a set of messages. Furthermore, the message system can provide a familiar and uniform interface to the Profile information, although particular software may want to have specially-tailored interfaction with it. Since folders are regular Unix files, such software need not go through the message system to access Profile information.

The following is a list of the features which are being provided; the major groupings (in capital letters) are according to the "messages" the user manipulates and the subordinate labels are the component names for the switches. The most common options for a switch are "yes", "no", or "ask". For the last alternative, the system each time asks the user if the option is to be performed each time possible; a few of these types of switches can only be yes or no.

NEW-MAIL:

Notify:

Whether to be notified of new mail; [yes/no].

Scan:

Should notification include a Scan display; [yes/no].

CREATION:

Signature:

Name to be used in the "From" component of messages created by the user; the exact text of this field is used as the signature.

Compose-contents:

Reply-contents:

Forward-contents:

These are intended to allow the user to tailor how the three functions create messages. In particular, what components to prompt for, what default fill-in text to place in components, whether to copy responses to other primary recipients, secondary recipients, and/or a personal file, and whether to provide feedback before sending a message. It is not yet clear how to have the user specify preferences. One thought is to use the RITA system (Anderson and Gillogly, 1976a; 1976b) developed at Rand, which is already intended for the construction of computer "agents" to act in the user's behalf.

Body-fill-in:

Whether always to format the body component of the draft, by filling and possibly justifying lines within paragraphs, as if the "Format" function had been invoked; [yes/no/ask].

COMPONENTS:

Equivalences:

Indicates equivalences between components, such as To and Action-to; a series of lists are used to indicate the equivalences. The lists are separated by semicolons or periods. For example: "To", "Action-to", "For"; "cc", "Secondary", "Info".

Address-lists:

Names of components to be treated as address lists; and which address component lists are to be shown in messages to all recipients, members of the same list, or not shown at all. Each component name is followed by the keyword "everyone", "members", or "authors," to indicate whether the text of that list is to appear in copies of the message sent to everyone, only other members of the address list, or only author(s) and the individual recipients respectively.

ALIASES:

Addressee aliases used during message creation. The name of a component contains the name to be typed by the user and the rest of the component contains the text that is to replace it. These aliases are only a typing convenience for individual users; the system-wide alias list, however, extends the number of "public" names.

TRANSFER:

Copy-display:

List-display:

Show-display:

The same use as for Compose-, Reply-, and Forward-contents, except that this controls the display, rather than acquisition of text for the indicated function; may also be viewed as "filtering out" parts of messages.

MISCELLANEOUS:

Alias-expansion:

Whether the system is to print out the expansion of personal aliases, at the time of their specification; [yes/no/ask].

COMMANDS:

Standard option settings to be used for individual commands. The name of a component is the name of the command and its contents are the standard settings. For example:

```
Scan:   recent
List:   -paginate -separate > listing-file
Format: -justify
```

mean that normally, the Scan function shows all recent messages; the List function paginates it output, starting each new message on a new page and places the listing in text file "listing-file"; and the the Format functional will normally right-justify text.

III. FUNCTION DEFINITIONS

For a summary of functions, see Appendix A.

The following is not a description of what is actually typed by the user, as there are several different human interfaces which are being constructed. The descriptions which follow are of the functions which will be available to users and of the vocabulary to be used in the command interface which approximately conforms to the Shell's syntax--see Appendix D for a description of that interface. The vocabulary is also believed to be appropriate for other interfaces.

These notational conventions are used for the following specifications:

(component)	=>	a single message component;
(components)	=>	a sequence of message components;
(draft)	=>	the draft message;
(file)	=>	a file name;
(msg)	=>	a single message;
(msgs)	=>	a sequence of messages;
( )	=>	other parameters, explained within text of particular descriptions;
[ ]	=>	optional information;
	=>	alternative specifications, one of which must be used.

Many functions change which message is the current one. In the following, descriptions indicate the rule for assigning the current message; no indication is made when the function does not affect the current message.

Except when a component is being modified through a text editor (i.e., Ned (Bilofsky, 1977) or Ed (Thompson & Ritchie, 1975)) text is always added to the end of components. This is done in a line-oriented manner; that is, the last character of a component is always an end-of-line, even if the appended text does not end with one.

Add (components)

The user is successively prompted for text, which is then Copied from the user's terminal to the end of each named component, in the draft. "Components" defaults to "Body".

Annotate (components) (msgs) (editor)

Allows modifying text in messages, while explicitly marking the modifications to the original text. The integrity of the original messages is thereby retained. The indicated text "editor" is

repeatedly called with the contents of the named components. The user may then make any changes described. When a component is returned to the system, it is automatically compared with the original form of the component and changes are surrounded with text marking them as annotations. The original versions of annotated components are saved in the draft backup folder. During implementation, various ways of marking changes are being tested. "Components" defaults to "Body" and "msgs" defaults to the current message. The last message in "msgs" becomes the current message, if it is not the draft.

Cleanup

Causes discarded messages to be expunged from the current folder, and discarded components to be expunged from the draft. For safety, command interfaces should require confirmation of this function, due to the impossibility of reversing its action. Note that no single function has been defined to perform a Cleanup and then automatically Quit, although most other message systems provide such a function. Cleanup is a sufficiently dangerous function that it should be completely isolated from other functions.

Also, it is planned that the remaining messages in the folder may be automatically sorted according to transmission date, author name or the like. More complete specification of this capability is deferred for the time being.

Compare (component) (msg) (component) (msg)

This is a generalization of the behavior described for the Annotate function. Text in the first component is compared with the text in the second component and differences are noted (in the first component). As with the Annotate function, the method for marking differences has not yet been determined.

Compose (components) (preserve)

Allows the user to enter text to the "To", "cc", "Subject", and "Body" components in draft. That is, the user is assisted in composing a simple message. If the draft already contains text, then the user is asked if a) it should be discarded, or b) if Compose should add onto the end of the text. At the end of the sequence, the user has the option of sending the message or returning to command level. If the draft is sent, it may be "preserved". "Components" alters the sequence of components for which text is prompted and facilitates creation of additional components. The system complains if the draft is not empty at the time this function is invoked and queries the user about proceeding.

Copy {(file) (component) / (msgs) (component) (name) / (msgs) (folder)}

This function provides a subset of the capabilities offered by the Map function. In particular, it is intended to facilitate performing the most frequently-used text transferring functions, without requiring the user to deal with the full complexity of the Map function.

The function's first alternative form allows copying the contents of a "clear text" file (one that is not a folder) to the end of a component of the draft message. The second option allows copying one or more existing messages onto the end of a component of the draft; "name" will cause the copied text to be prefaced with the name(s) of the source component(s); and the third option allows placing a copy of one or more messages, in the current folder, at the end of some other folder. In this last case, as with the third option of the Map function, the original messages are Marked as having been filed.

The Copy function is quite a bit more limited than the Map function. If the destination is a component, then it may only be in the draft. The source may be either an external file or else an entire message; selection of separate components is not allowed. The first alternative is like the Add function, except that the source of text is a file, rather than the user's terminal; and the third alternative is like the File function, except that the source messages are not Discarded. The last message in "msgs" becomes the current message, if it is not the draft.

Fig. 7 indicates the defaults used for each of the three options of Copy.

Correct (components) (msgs) (file)

Passes the named components through the Unix typographical-error detection program, which makes lists of possible spelling errors. A list is either placed into a component, in the associated message, which begins with the same name as the component being examined, but also has the suffix "-typos". Alternatively, the list may be placed into the indicated "file". "Components" defaults to "body" and "msgs" defaults to draft. The last message in "msgs" becomes the current message, if it is not the draft.

Describe (keyword)

This function is intended to allow the user to peruse online information about MS, while the Help and Syntax functions assume more urgency. A special message folder is searched for a message with a special component which contains the keyword and all the associated

Option	Source Message file	Destination Component folder	Comments
1	xx	body	
2	Current	xx	with (name) ?
3	Current	xx	

"xx" indicates that the parameter must be specified explicitly and may not be defaulted.

Fig. 7—Defaults for the Copy function

text is shown to the user. Note differences from the Help and Syntax functions.

Discard (components) (msgs)

Marks the indicated components as discarded from the message(s), but does not physically remove the text or re-order message numbering in the folder. This action is like placing a message in the wastebasket; it is still available, but somewhat less convenient to access, and is subject to permanent removal later, by the Cleanup function. Note that, as with other functions, this can be applied to the draft message. "Components" defaults to "All". "Msgs" defaults to the current message. A Discarded message is merely a message with all of its components deleted. If no Cleanup has been performed after a Discard, then the Retrieve function can be used to "un-discard" components, retrieving them from the "wastebasket" and placing them back on the "desk". The last message in "msgs" becomes the current message, if it is not the draft.

Ed (components) (msgs)

Repeatedly invokes the Unix Ed text editor (Thompson & Ritchie, 1975) with the contents of each named component. If the components are from old (non-draft) messages, the user is warned that the integrity of the messages may be compromised and the command interface usually requires confirmation. "Msg" defaults to "draft" and "components" defaults to "Body". The last message in "msgs" becomes the current message, if it is not the draft.

File (msgs) (folder)

Copies all components of the indicated "msgs" to the end of the named message file and then Discards them from the current file. "Msgs" defaults to the current message. The last message in "msgs" becomes the current message, if it is not the draft.

Format (components) (msgs) (justify)

Passes the named components through a fill-in/justify formatting program. The program causes blocks of text, separated by blank lines, to have lines filled-out with text, as close to the right margin as possible. "Components" defaults to "Body" and "msgs" defaults to "draft". "Justify" is a flag which determines whether text is to be right-justified or not. The default is not to justify, but this is of course settable in the Profile. This function is capable of being sufficiently traumatic that the previous version of the text is saved in the draft backup folder. The last message in "msgs" becomes the current message, if it is not the draft.

Forward (components) (msgs) (preserve)

Packages up existing message(s) for transmission to additional mail receivers. As with Compose, the draft is checked for existing text. Copies the "Subject" component, from the old messages, into the "Subject" component of the draft, bracketing each line. The resulting "Subject" component is displayed at the user's terminal. Allows the user to Add to the Body of the draft, if the user wishes to make comments about the text being forwarded; and then Copies the indicated components of the indicated messages into the Body of the draft, separating each message with some bracketing text: "--- Forwarded messages:" goes at the beginning, "--- End of forwarded messages" at the end, and a line of dashes in between messages. "Components" defaults to "All" and "msgs" defaults to the current message. "Preserve" is the same as for the Send function. The last message in "msgs" becomes the current message, if it is not the draft.

Goto (msgs)

The first message in "msgs" becomes the current message, if it is not the draft.

Help (keyword)

This is a primitive facility for providing online assistance. A special message folder is searched for a special component containing indicated text and the user is given text associated with the Summary and Syntax components of the Help messages. Note the difference from the Describe function. Calling this function with no parameters causes a general assistance message to be printed. Synonyms are allowed, to catch errors in terminology and typing, and they are pointed out to the user. The same kind of feature is provided in the initial user interface, to allow misnomers. One type of statistics gathering which the system will perform is of the incorrect command words chosen by users. These will later be added to the list of synonyms.

List (components) (msgs) (separate) (paginate) (heading) (file)

This is a primitive function for producing page-formatted sequential (e.g., hardcopy) output. The function creates a clear, sequential and "unprocessable" text copy of the named components. "Separate" indicates pagination between messages. "Paginate" indicates paginations within messages. "Heading" causes each page of output to be prefaced with the indicated text. When more than one message is Listed, a Scan listing is pre-pended. This function is not intended for producing text to be displayed on a CRT terminal, but rather for printing on a hardcopy device. "Components"

defaults to "All". "File" defaults to the text specified in the Profile. "Msgs" defaults to the current message. The last message in "msgs" becomes the current message, if it is not the draft. "Separate" and "paginate" are also defaulted.

```
Map {(file)              (components)(msgs)                    }
    {(components)(msgs)  (components)(msgs) (join) (name)      }
    {(components)(msgs)  (file/folder)      (join) (name)      }
                                                        (discard)
```

This is the basic text transferring function which can be used to:

1. Add text, from some file or from the user's terminal, into one or more components of one or more messages; (Option 1);

2. Add to existing components, or create new ones, based upon the contents of old components; (Option 2);

3. Transfer copies of components or entire messages to the end of other folders; (Option 3);

4. Transfer copies of components or entire messages to other types of files (i.e., "clear" text files); (Option 3, with "join" specified).

See also the Add, Copy, and File functions which offer tailored subsets of this function.

The name for this function is somewhat less predictable than the names given to other functions. Because of the function's generality and complexity, it is expected that users will not frequently employ it, so a name was chosen which would be likely to decrease the chances of a user's accidentally invoking it. User interface-builders, of course, may wish to use some other term; the word "map" is intended as a guide.

For the second and third alternatives of the function, the "discard" switch may be used to cause the original (i.e., the "source") copy of the transfered text to be discarded from the mailbox. For example, the File function uses the switch to give the appearance of "filing" the message, itself, into another mailbox.

As explained in the section describing the Text Transfer domain, the Map function uses information about the source and destination specifications to decide what kind of transfer to make. Four types of transfers are described above. A fifth can be distinguished by the use of the "join" switch with the second alternative. The primary unit of transfer is the component.

The first alternative is a transfer of sequential text, either from a file or from the user's terminal, added to the end of the destination components. Thus the user can include standard mailing lists to address components, prepared documents to the Body of the draft, and so on.

The second alternative also depends upon the "join" switch and whether the user indicates specific "components" for both the source and destination. If "join" is not set and only one list of components is specified, then the transfer is a map of those components from the source message(s) onto components of the same name in the destination message(s). If "join" is set or the user does specify both component lists, then the source components are joined into a block, as described for the fourth alternative, and added to each of the destination components. If "name" is set, then the names of the source components are added as prefatory text to the transferred string.

The third and fourth alternatives depend upon the "join" setting. Normally, the second alternative applies and the function creates a copy of a structured set of components (which thereby constitute a message) at the end of another folder; this action is equivalent to the third option for the Copy function.

If "join" is indicated, the fourth alternative applies; it is like the second alternative, except that the destination is an external file and not a structured message. The components are merged together, to form a non-structured, "clear-text" string of text which is then appended to the end of the indicated file. In this type of transfer, the copied text is no longer accessible as a message.

Fig. 8 indicates the ways that defaults are used to make specification more convenient. The first two entries for option 2 cause the same behavior; the user simply indicates the single component list differently.

Mark (components) (msgs) (status)

Alters the setting for the indicated status, such as "examined", "flagged", "answered", or "discarded". The Discard function is a special case of this function. MS is designed to allow easy addition of new status indicators. The last message in "msgs" becomes the current message, if it is not the draft.

Ned (components) (msg)

Same as Ed function, but invokes the Ned two-dimensional CRT editor (Bilofsky, 1977), which normally requires the user to have an Ann Arbor 40-line terminal. "Msgs" becomes the current message, if it is not the draft.

Option	Source Cmpnt	Source Msg file	Destination Cmpnt	Destination Msg file	Join	Comments
1		xx	body	draft		
2	xx	current	(<=)	draft	no	Components share meaning in these two
2	(=>)	current	xx	draft	no	
2	all	current	body	draft	xx	
3	all	current		xx	no	
4	all	current		xx	yes	
None	xx	xx				Need destination

"xx" indicates that the category of information has been specified explicitly by the user; "=>" and "<=" indicate that the component specification is the same as the one specified explicitly by the user.

Fig. 8 — Defaults for the Map function

Next

Shows the next message which is not discarded, relative to the current message. (Note the difference in meaning between this and the "next" message-reference keyword, in Figure 6.) Since a folder holds messages much like a file folder in an office, it is not possible to go to the "next" message after the last one; an error message is produced if this is attempted. The message shown becomes the current message.

Open (folder)

Switches primary attention to another folder. The Open function itself does not make any modifications to the original folder. When the system is first started, the user interfaces usually default to opening the user's inbox. However, they often can take an argument to cause the system to start with another folder. The basic system does not keep track of previously-opened folders, although interfaces may wish to, so that users can easily return to folders, without having to remember their names. Any new messages are incorporated into inbox each time it is opened. The default for this function is the user's inbox.

Previous

Shows the previous message which is not discarded, relative to the current message. (Note the difference in meaning between this and the "previous" message-reference keyword, in Figure 6.) Since a folder holds messages much like a file folder in an office, it is not possible to go to the "previous" message before the first one; an error message is produced if this is attempted. The message shown becomes the current message.

Process (components) (msgs) (program) (replace) (file)

Consecutively passes the named components to the named program. "Replace" indicates whether the output, produced from the processing, is to replace the original version of the components. If the components are not to be replaced, then the output is placed into components of the same messages which have names that are the concatenation of the original components' names and the "program" name. For example, Correct will normally place its output into "body-typo" in the draft. Correct uses Typo; Format uses Nroff. Alternatively, the output may be placed in a "file". If a component is replaced, then its original version is saved in the backup draft folder. The last message in "msgs" becomes the current message, if it is not the draft.

Quit

Causes the mail system to stop and returns the user to the calling program (usually Shell). Maintains enough information about a user's session to allow continuation of it when MS is run again. Notices when draft is not empty and notifies users (in case they forgot to send the message). This notification may become optional as determined by a Profile setting.

Reply (msgs) (recipients) (folder) (verify) (fcc) (preserve)

Facilitates sending a message in response to received messages. As with Compose, the draft is checked for existing text. The To component of the draft message is built from the From components of the indicated "msgs", the cc component is optionally built from address lists in the components named in the "recipients" parameter and from user input. If specified, the "fcc" component (file carbon copy) is set to be the "folder" specified or else to default to the user's inbox.

The Subject component of the draft is built from the Subject components of the indicated messages and, optionally, from user input. The text taken from the old messages is prefaced by "Re:"; to avoid a large number of nested brackets to occur, as a result of repeated replying, the preface is used only if one does not already exist, as when replying to a reply.

An In-Reply-To component is Added to the draft and contains the names (but not addresses) of the authors of the original messages, the dates (day and month) their messages were sent, and their message identification tags. This text is written in grammatical English.

After the standard components are created, their contents are displayed at the user's terminal, to allow verification. Then the user is allowed optionally to Add to the Subject and cc components and then to Add to the Body component of the draft. And finally, the message is optionally sent, as if a Send function had been invoked; and the old messages are marked as having been Answered. Other defaults are specified in the user's Profile. The verify switch is used to have the system request the user to "verify" inclusion of each potential recipient. And the "preserve" parameter is the same as for the Send function. "Recipients" defaults to the exclusion of all components; i.e., only the originator(s) will receive a copy. The last message in "msgs" becomes the current message, if it is not the draft.

Report

Allows users to send comments and complaints to the MS support staff. In reality, this function merely steps the user through a special Compose, creating an additional draft, and then automatically Sends the message to the appropriate people, including the report's author. A special draft, called "msreport", is maintained and is accessible in the same manner as the regular draft message. The user's regular draft message is not affected. Copies of reports are saved, in the draft backup folder.

Retrieve (components) (msgs)

The complement of the Discard function, which also works for the draft message. "Components" defaults to "All", "msgs" defaults to the current message. Computer users often call this an "undelete" function. The first message in "msgs" becomes the current message, if it is not the draft.

Revise (components) (msg) (editor)

This feature is intended to allow modifications to be made to existing messages, without explicitly indicating the strings of text which are changed. A separate component is used to record the fact of the modification. This latter component is like an audit trail. To a large extent, this function will be used when the reviser is violating the integrity of the original message but wishes to attribute original authorship. The function repeatedly invokes the indicated text "editor" on the named components. When revision is completed, a "Revision" component is Added to the message, with the user's name, the name of the revised component, and the date. If no "Revision" components currently exist for that message, then an "Originator" component is set to contain what was originally in the "From" component. The system therefore maintains an audit trail of modifications and preserves the name of the author of the message's original version. "Components" defaults to "Body" and "msg" defaults to the current message. Note that this function is not intended for use with the draft message, although such use is not prohibited. The last message in "msgs" becomes the current message, if it is not the draft.

Scan (msgs) (file)

Scans the messages and displays a "table of contents" listing of the indicated sequence of messages. The table includes folder index number, date sent, who from, the Subject component of the message, and indication of various aspects of each message's status. If the message contains no Subject or "Re" component, the initial portion of the message "Body" text (enough to complete the current line) is displayed. This text appears in the form

("this is the beginning...")

complete with parentheses, quotation marks and elipses. Given the current limitations of display format specification, this function cannot be defined in terms of a Copy or List. "Msgs" defaults to the group of recent messages. "File" defaults to the user's terminal.

Display format: SSS IIIC (LLLL) DDD From-Name Subject

Symbols:

SSS	Message's status (see below);
III	Message's index position in folder;
C	"<=" indicates the "current" message;
LLLL	Message length in lines;
DDD	The day and month of the message's Date component;
From-Name	Person's name or ID portion of the "From" component (sans hostname); and
Subject	As described above.

Only a portion of the possible status information is displayed with this function. For example, information about a message's having been answered or flagged is not included.

Status Indicators:

-	not seen
+	recent
*[	discarded

Send (preserve)

Packages up the draft message into a standard format and submits it for transmission. Contrary to most network message systems, MS attempts to send all mail immediately; users may choose to observe the process, but their choice does not affect the timing of transmission. Mail is actually queued for later transmission only when an initial attempt fails. A copy of the draft is filed into a backup folder, which is in the same directory as other standard MS files. Send may also be instructed to "preserve" the copy in draft.

Shell

This may be provided by the user interface and is not in the basic system. It is listed in this specification as a reminder of its utility for most interactive systems. The user is given access to a version of the Shell program (see Figure 2).

Show (components) (msgs)

Displays messages at the user's terminal. "Components" defaults to the set stored in the user's Profile (initially, "All"). "Msgs" defaults to the current message. The last message in "msgs" becomes the current message, if it is not the draft.

Statistics (name) (type) (value)

This function is not intended for the user; it is intended for standardized collection of user statistics, such as the names of functions that are called and the amount of computation which is required to perform particular functions. "Name" is an identification name which is unique to the caller of this function. "Type" is a sub-grouping identifier; and "value" is any text to be taken as a piece of data for this statistic. The actual usage of this function will conform to legal and social privacy considerations.

Syntax (keyword)

Displays the syntax for the indicated command. This function is a subset of the Help function, printing only the "Syntax" portion of the associated online assistance message.

?

This function is not in the basic system; it is recommended for inclusion in most user interfaces. The feature causes a display of the list of inputs valid at that level of a specification. Therefore this function is not intended just for top-level use. It should be possible to invoke it in any argument.

IV. STATUS OF THE IMPLEMENTATION

MS became partially operational at Rand in the fall of 1976. The "ms", "msg", and "mail" interfaces are all used regularly by Rand staff members. Distribution of the system to other ARPA project Unix machines was begun in late summer 1977. By that time, almost all of the originally-specified functions were built. Only Annotate and Compare have not yet been implemented. More seriously, no portion of the Profile exists; its lack is felt by all users, in particular for the purposes of regularly viewing only portions of messages and setting several switches to redefine the system's default actions.

In addition, the system does not allow blind carbon copies of messages and does not strictly enforce constraints on modifying Sender, Message-ID, and Timestamp. While specifying message addresses, users cannot yet include the contents of lists in files (with "<") or direct a copy to a folder (with ">"); address list names also are not properly handled. Their lack has not been seriously felt by users, at this stage of system use. The online assistance capabilities have been implemented only partially; and the Scan listing measures message length in number of characters and not lines. Users are notified of new mail only when they initially log into Unix and, when using MS, upon Opening their inbox. In a few cases, more general message and component selection capabilities (e.g., full Boolean) would have proved useful.

Current activities involve exporting the system to other sites, adding the Profile and increasing the efficiency of the system code. Portions of MS are currently quite slow and this has deterred some users from the system. The focus of this optimization effort is the parallel "structure" file which was initially implemented in an extremely general organization. Experience with MS has suggested a more constrained organization. It should be noted that the presence of a dual-file organization makes the transition between structures quite simple.

V. CONCLUSIONS

Although MS has so far received only limited distribution, current indications are that it successfuly fulfills its design goals. In particular, integrating access and modification capabilities with the draft and existing messages has proved extremely convenient. In general, the available functions and the style of their behaviors seem satisfactory to users, although the availability of the Profile would considerably improve some users' attitudes.

During the initial design review, the choice between using parallel files versus a single structured file led to some heated discussions. Experience to date thoroughly justifies the double-file choice, although its use did increase the complexity of the software needed to access and maintain text. The choice has meant that idiosyncratic, but necessary, modifications, such as massive reorganization to several messages, could be made to message files, without undue pain to the user.

The Map function has been a continuing problem. It has proved difficult to implement according to specifications and users are generally unable to employ it successfully. The Copy function is a direct result of these problems and it seems to adequately account for most users' needs, most of the time. It should be noted, however, that in at least one case a user wanted to copy a part of a message, into a draft component, and could not understand why the Copy function was unable to perform the function. This suggests that the focus on monolithicity is well-founded, and having the concept of the Map function has proved a useful focus for the MS project. In general, however, such per-component manipulation is not currently needed, though this may change as the Profile enables users to specify complicated actions once and then repeatedly re-use the specification.

The Shell-syntax interface to MS has variously encouraged and deterred new users. Some indicate that the similarity of syntax did, in fact, facilitate their learning to use the system; others indicate that the inherent complexity of the full MS domain requires more effort than they wish to expend. These users are quite comfortable with the msg interface. A confounding factor is the system's slowness. Some users are waiting to make the transition to MS until after it has been made more efficient.

Implementing the basic system at the level of user-functions, rather than the more common primitive-functions, has also been a mixed blessing. User interfaces are, in fact, easier to build and the extra software overhead of placing the higher-level functions into the kernel of MS, appears to be minimal. However, the communication discipline between the user interface and MS kernel system is not wholly adequate. In particular, the user interface cannot query the kernel for status information (e.g., whether a message is discarded) and cannot adequately select subsets of different functions' behaviors. Also, the kernel's interactions with the user, such as for verifica-

tion prior to performing some actions, cannot be fully controlled by the user interface. Remedies to these deficiencies are being considered.

From the standpoint of operational efficiency, it is unfortunately not currently possible to construct a simple system, with a subset of MS' full capabilities, without dragging along all of the software associated with the full system. The user need not see all of this, but it makes the programs more cumbersome. Some investigation is underway to discover how the system might be factored into smaller units; for example, infrequently-used functions, such as Cleanup, may be made separate processes.

Finally, use of the specification style led to a lack of precision in specifying the system's primitive functions. In some situations, this deficiency would have been disastrous. However, the project's operational environment made frequent consultation between members quite convenient. In addition, Bill Crosby, the system's primary implementor, usually chose to provide features in as general a fashion as possible; after experience was gained with the feature, tailoring it was usually quite simple. It should also be noted that much of the desired precision was not possible until we had that experience.

In spite of these problems, the specification style seems generally to have been useful, in that it has focused at the level of the user. Many systems, in spite of being examples of excellent software engineering, do not reflect this focus and are therefore inappropriate for most users.

Appendix A

SUMMARY OF FUNCTIONS

Add (component)

Sequential text is Copied from the user's terminal to the end of the named component in the draft.

Annotate (components) (msg)

Allows adding text to a message, while explicitly marking it as an addition to the original text. The integrity of the original message is thereby retained.

Cleanup

Causes discarded components and messages to be expunged from the message file.

Compose

Allows the user to conveniently Add to the "To", "cc", "Subject", and "Body" components in Draft, by prompting for their text.

Copy { (file) (component) | (msg) (component) | (msg) (folder) }

Allows copying the contents of files, or existing messages, into a component of the draft message, and copying entire messages to other message files.

Correct (component)

Passes the named component through the Typo spelling corrector program.

Describe (keyword)

For obtaining information about the message system. A special file is searched for the keyword and the associated text is shown to the user.

Discard (msgs)

Marks the indicated messages as deleted from the mailbox.

Ed (component)

Invokes Ed editor with the contents of the named component.

File (msgs) (folder)

Moves the indicated messages to the end of the named file.

Format (component)

Passes the named components through the Nroff text formatting program.

Forward (msgs)

Packages up existing messages for retransmission to other mail receivers.

Goto (msg)

Changes the current message to the specified message.

Help (keyword)

Primitive help facility. A special text file will be searched for the indicated text and the user will be given the initial text associated with the keyword.

List (msgs) (order) (options) (file)

This is a primitive formatting function for producing hardcopy versions of messages.

Map

The generic text-transferring function, which is inconvenient to use for standard transfers. See Add, Copy, and File function descriptions.

Ned (component)

Same as Ed function, but invokes the Ned two-dimensional CRT editor.

Next

Show the next message, relative to the current message.

Open (file)

Switches to another message file. When the system is first started, it defaults to opening the user's inbox.

Previous

Show the previous message, relative to the current message.

Process (component) (program)

Consecutively passes the named components through the named program. Correct uses Typo; Format uses Nroff.

Quit

Causes the mail system to stop and returns the user to the calling program (usually Shell).

Reply (msgs)

Allows responding to received messages.

Retrieve (msgs)

The complement of the Discard function.

Scan (msgs)

Scans the messages and produces a table of contents.

Send (preserve)

Packages up the draft message and submits it for transmission.

Show (msgs)

Displays the messages at the user's terminal.

Syntax (function)

Displays the syntax for the indicated function.

?

Displays a list of inputs valid at that level of a specification.

Appendix B

SAMPLE COMMAND INTERFACE

The sample command interface, specified here, is intended to be compatible with the syntax of the Unix Shell (see fig. 2); however, a few deviations are quite intentional.

In general, the user types the appropriate function name, to invoke a particular function. For convenience, the interface requires that only enough of the word be typed to distinguish it from other candidate names. For example "cop" means "copy". As an additional convenience, commands have a very terse form, which is shown immediately below the full form. A large number of synonyms have been defined for the commands and standard symbols, such as "examined". Users may type these synonyms, in place of the "official" terms, and they will be accepted, although they are not allowed to interfere with distinguishing between official terms. For example, "discarded" and "draft" are official terms referring to two different classes of messages; and "displayed" is a synonym for "seen". However, the user need type only "di" to mean "discarded" and must type at least "disp" to mean "displayed". The system is not so friendly as to advertise the synonyms it knows about. This limitation is imposed primarily to limit the length of listings produced with the ? function.

The system has a rudimentary error detection and correction facility appropriate to a line-oriented system. For example, upon detecting an error in part of a specification, the interface will notify the user of the nature of the error and prompt the user for the replacement information, saving all of the other information originally typed up to the point of the error. Except in the cases of folder and file names, the system will not make any distinction between upper and lower case characters in command lines.

The reader should remember that this interface is only one of several which are being implemented. It was the first interface built, in order to be compatible with the syntax of the existing Unix Shell, but is definitely not proffered as an example of a "friendly" human user interface. An MSG-type interface also is provided.

Defaults for function parameters are as recommended in the function descriptions. In addition, some abbreviated syntactic forms are allowed during specification; however, the interpretation of these depends upon context, as shown in the examples for the Copy Copy command, below. The "official" syntax, which conforms to Shell-syntax, does not have this dependency.

The system is invoked by typing "ms" to the Shell. A file name may be included as a parameter, in which case the indicated file, rather than the user's inbox, is opened.

The basic syntax for commands is:

```
command source -options > destination
```

where

command	is the command word;
source	is a filename or message/component specification;
options	are optional switch settings; each option ("switch") is prefaced by a dash ("-");
destination	is filename or message/component specification; ">" is required with destinations that are not defaulted.

Specific command descriptions indicate limitations on the above. Also, for prompted input from the terminal, such as for the compose function, the user may enter only one line of text (unless the last character is backslash, as shown below), unless a message is displayed to the user indicating that a Control-D (the ASCII EOT character) at the beginning of a line will terminate input.

Other standards, where applicable:

\	(Backslash, when preceding a carriage-return) Continue onto next line.
|	Passes the output to a process, rather than a file; in place of the ">" destination option.
--#	Where appropriate, means to reverse the the meaning of the indicated (#) switch; for example, in the Format function, "-j" means to right-justify text, so "--j" means that justification will not occur.
#	An integer, indicating a message's position within a mailbox.
#-#	A sequence of messages, starting with the first message and ending with the last.
-c	Following arguments are component references.
-f	Following argument is a file reference.
-m	Following arguments are message references.
x,x	The same as two arguments separated by space.
x x	Indicates a list of arguments, such as "to cc" or "3 4 7".
"..."	A quoted parameter, which allows the text to contain special characters such as space.

For convenience, the "-m", "-f", and "-c" switches often are not necessary. If the specification is a common one, then the text typed by the user will be interpreted correctly. For example, the formal specification for filing the current message into mailbox "filed.m" is "file -m current > -f filed.m"; however, the user actually need type only "f > filed.m".

Notational conventions, for the following descriptions:

c	A single component may be referenced at this point;
cs	Reference to a number of components is legal;
f	Reference to a file is allowed;
m	Reference to a single message is allowed;
ms	Reference to a number of messages is allowed;
( )	Other information may be specified; the nature of this information is explained in the text of the associated description.

Commands

{Add / A} c

{Annotate / An} cs/ms [-e]

-e Following argument names text editor

{Cleanup / Cl}

{Compare / Cpr} c/m > c/m

{Compose / C} [-c] [-p]

-p Same as Preserve option, for Send

{Copy / Cp} { f > c / ms > c [-n] / ms > f }

-n Indicates that component names are to preface component text, when the second specification option is used.

Examples

Cp > backup.msg
Appends the current message to the end of file "backup.msg".

Cp -m 2 > -c Body
Adds message 2, as a block, to the end of the Body of the Draft. The "-m" is gratuitous, but the "-c" is not, since the destination of a message is usually a file.

Cp -f document > body
Appends the contents of file "document" to the end of the Body of the draft. The "> body" is gratuitous, since text copied from a file usually goes to the body of the draft. However, since the source of text is usually a message in the current mailbox, the "-f" specification is necessary.

{ Correct } cs/m
{ Crct }

{ Describe } (keyword)
{ Dsc }

{ Discard } cs/m
{ D }

Ed cs/m

{ File } m > f
{ F }

{ Format } cs/ms [-j]
{ Fm }

-j Justify

{ Forward } cs/ms [-p]
{ Fw }

-p Preserve draft, as in Send

{ Goto } m
{ G }

{ Help } (text)
{ H }

{ List } cs/ms [-h] [-p] [-s] > f
{ L }

-h Use next argument as page header;
-p Paginate within messages;
-s Separate messages; start each one on a

new page;

Example

L 3-9 -o memoform -p -h "Noteworthy Stuff" | lpg
Will list messages 3-9 on the printer; listing will be paginated with the indicated heading, and components will be ordered according to the list in the profile called Memoform. Messages will not begin on a new page.

{Map} { f > cs/ms } [-d]
{ cs/ms > cs/ms [-j] [-n] }
{ cs/ms > f [-j] [-n] }

-d Discard source version of text;
-j Turns on the join switch;
-n Indicates that component names are to preface component text, when the second specification option is used.

Examples

Map > backup.msg
Appends the current message to the end of file "backup.msg".

Map -c Subject -m 2,5,9 > -c Subject Keyword
Appends the text of the Subject components in messages 2, 5, and 9 to the Subject component and then the Keyword component of the draft.

Map -m 2 -c Subject,From CC To > -c Body
Adds the source components as a block to the end of the Body.

Map -m 2-5 -c From To CC BCC > -m 9
Adds the contents of each of the indicated address components onto components of the same name (creating them if they do not already exist) in message number 9.

Map -m 2-5 -c From To CC BCC > -m 9-10 -c From Subj
As when the text was copied to Body, above, the text is copied as a single group but to the end of the From and then the Subject components of messages 9 and then 10.

Map -m 2-5 -c From To,CC,BCC -n > -m 9-10 -c From Subj
Same as above, except that the text of each source component is prefaced by its component name.

{Mark} cs/ms (status)
{Mk }

{Ned cs/ms}

{Next / N}

{Open / O} f

{Previous / P}

{Process / Prc} cs/ms (program) [-r]

-r replace each component with the output of the processing.

{Quit / eot (control-D) and Q}

{Reply / Rpl} ms [-a] [-i] [-p] [-v]

-a Author copy: Place "inbox" into fcc component.
-i Copy contents of the components, named in the following parameters, into the cc component of the draft.
-p Preserve, as with Send.
-v Verify inclusion of each addressee.

{Report / Rp}

{Retrieve / R} cs/ms

{Revise / Rv} cs/ms [-e]

-e Following argument names editor

{Scan / Sc} ms [>f]

{Send / Snd} [-p] [-q] [-s]

-p Preserve Draft after sending.
-q Queue mail.
-s Send mail immediately.

{Shell / Sh}

{Show / S} cs/ms

{Syntax / Sy} (command)

?

Appendix C

NON-EXISTENT OR DISCARDED TEXT

During the initial phases of implementation, a question arose concerning the way in which MS should deal with user references to discarded or non-existent text. An exhaustive list of behaviors was created. It is included here because it represents a statement of philosophy concerning the treatment of user errors. What did the user probably mean? Some references are completely specific, in which case the user probably believes that the message is not discarded and therefore probably needs to be told, or when safe, the action should be performed. In other cases, an implicit reference is made, such as "examined", in which case the user probably does not care that a few "extra" messages have been included; so the user is not burdened with the information that s/he has made an error.

In the following table,

Yes/No	indicates whether the function is performed, or not;
Note/Quiet	indicates whether a notice is displayed to user, or not;
Replace	indicates whether discarded text is replaced; and
Flag	indicates whether individual discarded messages are noted.

The Show function distinguishes between specific reference, as in "show 3" and implied reference, as in "show all".

Function	Component reference	Msg reference
Add	Yes; quiet,replace	Not applicable
Comment	No; note	Not applicable
Copy	-- See Map	--
Correct	No; note	Not applicable
Discard	No; quiet	No; quiet
Ed/Re	Yes; replace	Not applicable
File	Not applicable	No; note
Format	No; note	Not applicable
Forward	No; note	Yes; quiet
Goto	Not applicable	Yes; Discard: quiet Not exist: note
List	No; quiet	No; note
Map	Srce: no; quiet Dest: note & replace	No; note
Next/Previous	Not applicable	See Goto and Show/implied
Process	No; note	Not applicable
Reply	Not applicable	Yes; discard: quiet not exist: note
Retrieve	No; quiet	Yes; quiet
Revise	No; note	Not applicable
Scan	Not applicable	Yes; flag
Send	Not applicable	No; note
Show/implied	No; quiet	No; note
Show/specific	Yes; flag	Discard: yes, flag; Not exist: no; note

Appendix D

A COMMAND INTERFACE SCENARIO

Text typed by the user is in boldface. Comments on the scenario are italicized on the right-hand side of the page.

% ms *Start the system.*

MS: 11 Jan 77
Incorporating new mail . . . *New mail is automatically added and is scanned.*
Folder inbox has 36 messages in it.
2 messages have not been examined yet.

+ - 35 (176) 31 May To: Dcrocker Exciting example of MS session
+ - 36 (358) 31 May To: Dcrocker Pity the poor reader Re: exciti

–> n *User can step thru looking at new mail*

(Message 35, 176 bytes)
Date: 31 May 1977 at 1725-PDT
From: dcrocker at Rand-Unix
Subject: Exciting example of MS session
To: dcrocker

It is a little strange sending myself a message.

Dave

–> n
(Message 36, 358 bytes)
Date: 31 May 1977 at 1727-PDT
Subject: Pity the poor reader
Re: Exciting example of MS session
From: Dcrocker at Rand-Unix
To: Dcrocker at Rand-Unix
cc: Dcrocker at Rand-Unix
Message-ID: [Rand-Unix]31-May-77 17:27:15. Dcrocker
In-Reply-To: Your Message of 31 May 1977 at 1725-PDT

Not only is it strange, but it is also likely to be very dull. D/

And can then do other work.

–> sc 10-14

10 (637) 24 May To: Jim Re: [report] Adding standard
11 (624) 24 May Jim Re: [report] Adding standard
12 (1046) 24 May To: Jim Re: [report] Adding standard
13 (680) 25 May Jim Re: [report] Adding standard
14 (593) 26 May To: Jim Unadvertised MS feature

–> file 10-14 > /rnd/dcrocker/gnome/reports/jim

A series of messages worth saving; (User likes long names.)

–> o /rnd/dcrocker/gnome/reports/jim
Folder /rnd/dcrocker/gnome/reports/jim has 16 messages in it.

User then shifts attention to a new folder.

–> sc all
1 <= (705) 5 May Jim [report] Seek error

Checks what messages are there.

.
.
.
6 (383) 24 May Jim [report] Adding standard fields
7 (471) 25 May Wec Re: report: strange field
8 (639) 25 May Wec Re: [report] Adding standard field
.
.
.

–> s6

Notices a pending item and reviews.

(Message 6. 383 bytes)
Date: 24 May 1977 at 0727-PDT
Subject: [report]
Adding standard fields
From: Jim at Rand-Unix
To: wec at Rand-Unix, greep at Rand-Unix, dcrocker at Rand-Unix
cc: Jim at Rand-Unix
Message-ID: <[Rand-Unix]24 May-77 07:27:08.Jim>
MS-Version: 11 Jan 77

Can one add standard system-filled fields (like Date and Message-ID) at the interface level, or is that a change in msgsys.c?

–> show 8
(Message 8, 639 bytes)
Date: 25 May 1977 at 1019=PDT
Subject: Re: [report]
Adding standard fields
From: Wec at Rand-Unix
To : Jim at Rand-Unix
cc: Dcrocker at Rand-Unix, Wec at Rand-Unix, Greep at Rand-Unix
Message-ID: <[Rand-Unix]25-May-77 10:19:57.Wec>
In-Reply-To: Your Message of 24 May 1977 at 1129 PDT
<[Rand-Unix]24-May-77 11:29:30.Jim>

Although the message system will allow the user interface as well as the user to create any arbitrary fields it is not easy for the user interface to then put data into them. Were you planning on doing this sort of operation? If so then maybe we should work out a simpler mechanism.

Bill

```
-> rp 8 ?
- Message --
Number
Number-Number
        all
        beginning
        current
[not] discarded
        draft
[not]   new
        old
[not]   recent
[not]   replied
[not]   seen
[ -m msg-seq ] [- c comp-seq]
-i include-list
-p[reserve]
-v[verify]
```

And then decides to send a note to everyone but can't remember how to include them. The system prints a list of inputs which are valid, here.

```
-> [reply 8] - i to,cc

Draft already exists.
OK to discard it and continue? no
OK to continue and add to it? no
Compose aborted.
-> s dr
To: Wec at Rand-Unix
cc: Jim at Rand-Unix, Dcrocker at Rand-Unix, Wec at Rand-Unix,

-> discard draft

-> rp 8 -i to,cc
Subject:    Interface-generated input to components
To:   Wec at Rand-Unix
cc:   Jim at Rand-Unix, Dcrocker at Rand-Unix, Wec at Rand-Unix,
      Greep at Rand-Unix
Subject:    Interface-generated input to components
            Re: [report]
            Adding standard fields

cc:
Input body. End with <return> <control-D>.
Let's plan to have a meeting next week to discuss the best way to provide this feature. OK?  Dave.

Do you want to send the message now?  yes
Message being processed.
Processing completed and Draft discarded.

-> o

-> sc dis
*[      17      (317)   20   May  Jim          Additional standard fields ]
*[      21      (637)   24   May  To:  Jim     Re: [report]  Adding standard ]
*[      22      (624)   24   May  Jim          Re: [report]  Adding standard ]
*[      23      (1046)  24   May  To:  Jim     Re: [report]  Adding stand ]
*[      28      (680)   25   May  Jim          Re: [report]  Adding standard ]
*[      31      (593)   26   May  To:  Jim     Unadvertised MS feature ]
*[      33      (239)   30   May  To:  wec     first/beginning and last ]
*[      34      (332)   30   May  To:  Dcrocker samples for a document]
```

And then reprompts user, indicating what has already been specified. Then system notes that Draft is not empty and queries user.

User checks Draft

Decides to erase it

Then start reply over.

User sends note

Then returns to inbox

And performs some housekeeping

```
*[      34      (332)   30   May  To:  Dcrocker samples for a document
                                                              Re:  te]

-> cl                                          Decides to expunge discarded messages.
-> o

Folder inbox has 28 messages in it.
-> quit                                        then exits MS
%                                              and is returned to the unix shell
```

REFERENCES

Anderson, R.H. and J.J. Gillogly, Rand Intelligent Terminal Agent (RITA): Design Philosophy, R-1809-ARPA, The Rand Corporation, Santa Monica, California, February 1976a.

Anderson, R.H. and J.J. Gillogly, "The RAND Intelligent Terminal Agent (RITA) as a Network Access Aid," Proceedings of the 1976 National Computer Conference, 1976b.

Bilofsky, W., The CRT Text Editor Ned: Introduction and Reference Manual, R-2176-ARPA, The Rand Corporation, Santa Monica, California, in preparation, 1977.

Bobrow, D.G., J.D. Burchfiel, D.L. Murphy, and R.S. Tomlinson, TENEX, A Paged Time Sharing System for the PDP-10, BBN Report No. 2180, Bolt Beranek and Newman, Cambridge, 1971.

Broos, M.S., E.H. Black, and A. Vezza, MSGDMS Manual (draft), Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, 1975.

Brown, T. and M. Klerer, "The Effect of Language Design on Time Sharing Operational Efficiency," International Journal of Man-Machine Studies, Vol. 7, 1975, pp. 233-247.

Carbonnel, J.R., J.E. Elkind, and R.S. Nickerson, "The Psychological Importance of Time in a Time Sharing System," Human Factors, Vol. 10, No. 2, 1968, pp. 135-142.

Carlisle, J.H. "Man-Computer Interactive Problem Solving: Relationship Between User Characteristics and Interface Complexity," Ph.D. Dissertation, NTIS No. AD786466, School of Organization and Management, Yale University, New Haven, June 1974.

Card, S.K., T.P. Moran and A. Newell, The Manuscript Editing Task: A Routine Cognitive Skill, PARC Report No. P76-00082, Xerox Systems Science Laboratory, Palo Alto Research Center, Palo Alto, California, 1977.

Crocker, S.D., J. Heafner, R. Metcalfe, and J. Postel, "Function-oriented Protocols for the ARPA Computer Network, Spring Joint Computer Conference, Vol. 40, 1972, pp. 271-279.

Engelbart, D.C., Coordinated Information Services Discipline- or Mission-Oriented Community, Network Information Center No. 12445, Augmentation Research Center, Stanford Research Institute, Palo Alto, California, 1972.

Heafner, J.H. "Design of Application-Oriented Languages by Protocol Analysis," Ph.D. Dissertation, University of Southern California, Los Angeles, 1976.

Heafner, J.H. and L.H. Miller, Design Considerations for a Computerized Message Service Based on Triservice Operations Personnel at CINCPAC Headquarters, Camp Smith, Oahu, ISI/WP-3, Information Sciences Institute, University of Southern California, Marina del Rey, California, September 1976.

Miller, George, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," Psychological Review, Vol. 63, 1956, 81-97.

Myer, T.H., J.R. Barnaby, and W.K. Plummer, TENEX Executive Language Manual for Users, Bolt Beranek and Newman, Cambridge, 1971.

Myer, T.H. and D.A. Henderson, "Message Transmission Protocol," Arpanet Request for Comments, No. 680, Network Information Center No. 32116; Augmentation Research Center, Stanford Research Institute, Menlo Park, California, 1975.

Myer, T.H. and C.D. Mooers, Hermes Users' Guide, Bolt Beranek and Newman, Cambridge, 1976.

Panko, R., The Outlook for Computer Message Services: A Preliminary Assessment, Telecommunication Sciences Center, Stanford Research Institute, Palo Alto, California, March 1976.

Pogran, K., J. Vittal, A. Henderson, and D. Crocker, "Proposed Official Standard for the Format of ARPA Network Message Headers," Arpanet Request for Comments, No. 724, Network Information Center 37435; Augmentation Research Center, Stanford Research Institute, Menlo Park, California, May 1977.

Ritchie, D.M. and K. Thompson, "The UNIX Time-sharing System," Communications of the Association for Computing Machinery, Vol. 17, No. 7, July 1974, pp. 365-375.

Roberts, L. "Computer Network Development to Achieve Resource Sharing," Spring Joint Computer Conference, Vol. 36, 1970, pp. 543-549.

Thompson, K. and D.M. Ritchie, Unix Programmer's Manual, Bell Laboratories, Murray Hill, New Jersey, 1975.

Tugender, R. and D.R. Oestreicher, Basic Functional Capabilities for a Military Message Processing Service, Information Sciences Institute, Marina del Rey, California, 1975.

Uhlig, R., "Human Factors in Computer Message Systems," Datamation, Vol. 23, No. 5, May 1977, pp. 120-126.

Vezza, A., "A Model for an Electronic Postal System," In B.M. Owen (ed.), Telecommunications Policy Research Conference Proceedings. Aspen Institute Program on Communications and Society, 360 Bryant St., Palo Alto, California 94301, 1976.

Vezza, A. and M.S. Broos, "An Electronic Message System: Where Does It Fit?" In Trends and Applications 1976: Computer Networks, Institute of Electrical and Electronic Engineers, New York, 1976.

Vittal, J. MSG Users Guide, Information Sciences Institute, Unversity of Southern California, Los Angeles, 1975.

Walther, G.H., The Online User-Computer Inteface: The Effects of Interface Flexibility, Experience, and Terminal-Type on User Satisfaction and Performance, Ph.D. Dissertation, NTIS No. AD777314, University of Texas, Austin, 1973.

Yntema, D.B. "Keeping Track of Several Things at Once," Human Factors, Vol. 5, 1963, 7-17.

Yntema, D.B. and G.E. Meuser, "Remembering the Present State of a Number of Variables," Journal of Experimental Psychology, Vol. 60, 1960, 18-22.

Yntema, D.B. and G.E. Meuser, "Keeping Track of Variables that Have Few of Many States," Journal of Experimental Psychology, Vol.63, No. 4, 1962, 391-395.

Yntema, D.B. and G.M. Schulman, "Response Selection in Keeping Track of Several Things at Once," Acta Psychologica, Vol. 27, 1967, pp. 316-324.

Yonke, M. BANANARD Users Guide, Information Sciences Institute, University of Southern California, Los Angeles, 1975.